# 1:24000+ Scale Hydro Clearinghouse
## Enhancements

With any system, there are opportunities for improvement or desired enhancements. The following list is provided to assist REO in planning for ongoing improvement of the Clearinghouse server.

- Additional quality assurance tests. This version of the server and client comes with basic QA to determine that all coverages are present, and that they have the correct precision, projection, and topology. Additionally, the attribute tables, relate tables, and event tables must have the correct layout and item definitions. The QA tests further verify that features all fall completely inside the edit boundary polygon. Furthermore, it determines that all LLID values are unique within the checkout polygon, and the positions represented by decoded LLID values all fall inside the checkout polygon. A framework for adding additional QA routines is provided. Numerous QA AMLs are available from the hydro partners, including Washington DNR (Travis Butcher, Washington Dept of Ecology (Dan Saul), REO (http://www.reo.gov/hydro/cgm) and others. The code for many of these tests has already been written. Two approaches can be taken for incorporating these programs: Standardize them into the clearinghouse AML system, or simply run them and incorporate their reports by concatenating them to the main report. The first approach would result in a more standardized report that would be easier to interpret. The second method would be cheaper to implement, and easier to incorporate updates from contributing agencies. QA enhancement requests are:

- QA requiring the comparison of “before” and “after” coverages.

    - Verify that no streams were deleted in the editing process. This request is from the BLM (Dan Wickwire). It has not been determined if the same test is required for the other hydro layers.

    - Verify that the route measures at stream confluences have not shifted during editing. This requirement is optional for conversion from 1:100000 data to 1:24000+ data, but it is not optional for the editing of 1:24000+ scale data. An untested AML has been written by Washington DNR (Charlie Ware) to do this test. It is named CHECK_MEAS.AML. A copy has been included in the AML directory.

    - Verify that section event tables extracted from the clearinghouse cover all portions of routes to be extracted. This test would be run on the server only. If the stored section event tables are incomplete for some reason, routes or portions of routes would be lost in the extracted data. Routes are rebuilt from section events. Any routes or portions of routes not covered by section events would not get rebuilt. Two cases that could result in missing section events are:

        - Route editing or creation without using the clearinghouse server application to check in or check out data.

        - A bug or crash of the clearinghouse server application that results in section event records not being stored.

    - QA that can be done on a single coverage. Verify that:

        - All code values are present in the corresponding lookup tables
        - All required fields are populated
        - All required event records are present
        - Features edgematch to the background coverage
        - No watercourse (WC) routes cross each other except canals
        - Braids may touch at the upstream ends, but they do not cross
        - The downstream end of each watercourse route must connect cleanly. No over- or under- shoots, within a tolerance.
        - All watercourse routes have ascending measures upstream
        - All route measures are unique within shoreline (WS) and watercourse (WS) routes, and monotonically increasing
        - Events fall at least partly on a route. An event may have a measure value of 9999, meaning that the event goes to end of a route, even if the route is extended later.
        - There is at least one metadata event for each route, each time it is edited
        - Waterbody (WB) areas must be single-part and non-overlapping
        - Shoreline (WS) routes having a default shoreline code of "Y" must be coincident with the boundary of one or more waterbody regions
        - Each waterbody region must be routed with shoreline routes having a default shoreline code of "Y". These routes cannot overlap with other default shoreline routes.
        - There is no restriction on the placement of shoreline routes having a default shoreline code of "N"

- Extensive documentation. This system comes with a system design, installation, and operation document, help files, and code comments. There are other forms of documentation that may be useful:
    - Data submittal guide. During the data design meetings, the need for a data submittal guide was discussed to help participants create data that adheres to the standards. This guide would include such topics as routing rules, feature-level metadata standards, etc
    - User guide for the QA AMLs, web interface, and FTP client.
    - System operations manual
    - Process flow diagrams
    - System architecture diagrams
    - Program architecture diagrams
    - Detailed installation guide

- Bulletproof the server. The server logs all significant events both to transaction tables and to a log file. It keeps copies of all extracted and submitted data. It stores the data in Oracle. It uses a temporary ArcSDE version during checkin to avoid creating incomplete and corrupt data during a failed checkin. Server errors are reported to the clearinghouse manager via email. Still, more bulletproofing can be done. The ChseServer class has two methods named BeginTrans and EndTrans. These methods are called before and after each major transaction processing step. Currently, these methods do nothing. The plan is that BeginTrans will create a persistent record that a transaction step has been started, and EndTrans will remove this persistent record when the step finishes. When the server starts up, it will check for a crash by looking for data created by BeginTrans. If it finds this data, it will know the transaction number and transaction step that failed. It will enter recovery mode and redo or undo the step that failed. After coding these methods, the server should be tested by deliberately crashing it during various transaction steps to verify that it can successfully recover.

- Enhanced security features.
  - Each transaction comes with a validation code. This code is a random number which could be used to ensure that data submitted to the incoming FTP site actually originated with the owner of the transaction. Currently, only the transaction id is used for this purpose, and the transaction ids are public. To implement this feature, the incoming transactions would be required to have the validation code in their readme.txt files.
  - Encrypt passwords. Currently, editor passwords are kept in an unencrytped column in the EDITORS table. These passwords are only used by the Clearinghouse server. They are not OS or database passwords, so they do not represent a security hole in that regard. The risk is that someone could masquerade as a data editor if they got a password. They would probably be detected quickly, as Email is sent to each editor whenever they check in or check out data.

- Data browsing website. The web pages delivered with this application have limited usefulness to people who are not involved in the editing process. A website that showcases the clearinghouse data could be a valuable contribution to data dissemination by REO as a node in the Geography Network.
  - Event data display. Display of event data at this website is also an enhancement to be considered. A method of pre-calculating event locations would make events display quickly.
  - Data download server. Various approaches could be used here, including nightly exports of the data, either as a single huge export file, or broken down by watershed. An on-demand database query and sownload would also be an option here. Some users may want to download a copy of some data for a project, but there is a danger here that they might try to submit changes to this data, thus potentially stepping on someone else's edits. This "readonly" web site should be carefully positioned to data editors so they do not "end run" the transaction process by editing data they copied from this "readonly" web site.
  - Some of these "readonly" pages would be useful to editors as well as browsers:

  - Checked-out areas
  - Query of editing history by location – just the checkout boundary polygons
  - Note about feature history: We are currently keeping a separate version for each completed transaction. These versions can be used to display historical data. As these transactions can have a lot of data, the system performance may degrade over time, necessitating a policy change whereby old versions are deleted. Therefore, it would be unwise to promise the ability to display and query historical data until the system has been in operation long enough to determine the feasibility.

- A feature to detect non-unique LLIDs even if they are inaccurately placed. The system currently checks for unique LLIDs by determining if the LLIDs are unique within the checkout polygon, and verifying that each LLID is derived from a Longitude, Latitude value that falls inside the checkout polygon. This method works if the data are perfect. An enhancement is proposed to make it work with imperfect data. Two forms of imperfect data are anticipated:
  - 1. LLIDs in the 1:100,000 data were generated from features in the NAD27 datum, even though the features themselves are stored in Decimal Degrees in the NAD83 datum. In this clearinghouse server implementation, this was used as the standard, so all LLIDs are projected from NAD27 to a temporary point coverage in the NAD83 datum for testing to verify that they fall inside the checkout polygon. This projection results in a shift of about 100 meters. If data are submitted with LLIDs generated in the NAD83 datum, they may fall outside the checkout polygon in some cases, resulting in the detection of a spurious data error by the server. If LLIDs of features in bordering data describe points that fall inside the checkout polygon, there is a small probability that a legitimate feature created inside the checkout polygon will have this same LLID, thus resulting in a condition of non-unique LLIDs in the database.
  - 2. As data are conflated from 1:100000 to 1:24000+, a feature may move away from the point represented by its LLID. The same alpha and beta errors described above may occur in this case.
- A solution to this problem could involve the checking out of a buffer zone around the checkout polygon, and the generation of a separate list of LLIDs for features in this buffer zone. No features could be added that fall inside this buffer zone. It would be for LLID uniqueness only. LLIDs would need to:
  - Be within a tolerance of their respective features.
  - Be unique within the checkout polygon AND the buffer zone.

- Eliminate the need for the storage of section event tables. The system currently stores the sections of the two route layers (WS and WC) as events. This is done so that the same sections will result in data checked out as existed on the data that were checked in. The default condition is that a separate section results between each pair of vertices. Attempts to dissolve sections in the coverage data model will not work without using these section event tables due to precision issues. Routes longer than 1000 km are anticipated with vertex spacing as close as 1 m are anticipated. Measures are stored in the coverage as single precision floating point numbers. Any reasonable “length-measure ratio” section dissolve tolerance on this data would be subject to too much rounding error to be

effective. The measures are stored much more precisely in ArcSDE, however. The section breakpoints could be derived in ArcSDE, before the coverage is exported. A section event table could be derived on the fly as the data are extracted, then used the same way section event tables are currently used to process output data. The advantage here is that geodatabase tools could be used to edit routes directly. Under the current system, only coverage editing can be done, because of the need to preserve section breakpoints as section event tables.

- Submit planned checkin date at checkout time, and implement a system of email reminders. Currently, the checkout date and a deadline date are stored. The deadline is calculated as checkout date + one week. The two specific enhancements are a flexible system for assigning deadlines, and a system of email reminders for overdue transactions.

- Add a comment box on checkout polygons so people can query to find out why an edit is being done. This would require the addition of a comment field to the transaction table, modification of the three user interfaces that create transactions, and an enhancement to the web interface for displaying checked out areas. This enhancement was requested by Andrew Kinney of the Thurston Geodata Center.

- Set up a system whereby stakeholders can be automatically notified by email when transaction activity occurs in their areas of interest. This could be a useful tool for multiple data custodians to keep track of activity in their overlapping areas of responsibility. This could be implemented with a multipart polygon layer with 1 row for each person to be notified. The multipart polygon would cover their area of interest. Each major transaction event such as checkout, data submission, transaction post, or transaction abort would generate email when it happened in their area of interest. It would probably be simplest to join this layer to the EDITORS table, and add a field indicating the editor's role (editor, data custodian, interested party, etc.) only certain roles could actually edit data. This enhancement was requested by Diane Ransford of the Siauislaw national forest.

- Add the ability to check out portions of long routes that fall outside the checkout polygon. Currently, all features to be edited must fall inside the checkout polygon. To edit a small area with a 1000 km route passing through it, the user has two choices:
  - Don't edit it
  - Create a checkout polygon that includes the area of interest with long, skinny protrusions that surrounds the long route. This polygon could be created by buffering the long river, and merging the buffer polygon with the area of interest polygon. The resulting checkout polygon would resemble a snake that ate a rabbit.

  One proposed method is to split features that cross the checkout polygon, give the parts to different editors, and merge the parts when the data are checked in. This method will take a considerable amount of programming effort, because it lowers transaction granularity from the feature level to the sub-feature level. It also poses unsolved technical issues. For example, what happens to the events when two people are editing the same long route? Events often span long portions of routes, and changes to event

measures can cause the events to change position along the route, sliding them into the other editor's territory. Also, it is unclear how to assign events to different editors using this method. For example, WC_EVT_NAME typically spans the entire route. In a split feature scenario, which editor is allowed to edit this event? It would be very difficult to allow two people to work on the same long route without the possibility of edits to the measures causing conflicts on the event the data, and it is unclear how events spanning long reaches of the route can be shared between different editors.

Here is an alternate approach for long route management that only allows one editor on each route: The user checks out a reasonably sized polygonal area of data using the standard method. Then, in the same transaction, the user types in the LLIDs of one or more long routes that pass through the polygonal checkout area. These LLIDs are added to a list of a special checked out long routes, so that nobody else will edit them. This list of LLIDs and user IDs of people editing them would be maintained on the server as an Oracle table. When the user checks in his data, an integrity check is done to verify that no geometry changes were made to the long route except where the route falls inside the checkout polygon. Thus, features edited in other transactions can still snap to the long route in a background coverage. The user can make any event changes necessary. This appears to be the only practical way to guarantee that editors do not conflict with each other on a long route without checking out a polygonal area that completely surrounds the route. It works because only one person is editing a route at a time, and they are not making geometry changes that would cause snapping problems for other editors.